Safety Concept for a Mixed Criticality On-Board Software System

Alejandro Alonso Juan A. de la Puente Juan Zamorano
Miguel A. de Miguel * Emilio Salazar Jorge Garrido

Abstract: This paper presents a safety concept for the on-board software system of the UPMSat-2 experimental satellite. Subsystems with different criticality levels are assigned to different partitions in a partitioned software architecture. The architecture is based on XtratuM, an open-source virtualization kernel, and ORK+, an open-source real-time kernel for high-integrity systems. The safety requirements of the system are analysed, and a safety concept is developed based on the partitioned software architecture. The properties of the implementation resulting from the approach are discussed, and guidelines for future developments are proposed.

Keywords: Computers in control, real-time systems, mixed-criticality systems.

1. INTRODUCTION

Modern embedded systems typically integrate a multitude of functionalities with potentially different criticality levels. In some domains, systems with high criticality components have to be certified or qualified according to some domain-specific standard, e.g. IEC 61508 for electronic systems, DO-178B/C for avionics, IEC 880 for nuclear power plants, EN 50128 for European railway systems, or the ECSS series for European space systems.

As embedded systems are becoming more and more complex, the integration of mixed-criticality subsystems can lead to a significant and potentially unacceptable increase of certification efforts. One approach to limit such effort is to incorporate mechanisms that establish multiple partitions with strict temporal and spatial separation (TSP) between them. Under this approach, subsystems with different levels of criticality can be placed in different partitions and can be verified and validated in isolation.

This paper presents a safety concept for a space on-board software system based on results from the MultiPARTES [1] and HI-PARTES [2] projects. The techniques used are illustrated by a case study of an experimental satellite system. The software runs on a multicore hardware platform, on top of which several software partitions are defined in order to execute subsystems with different levels of criticality.

The paper is organised as follows. The characteristics of the satellite system and the general structure of the on-board software are described in section 2. Section 3 reviews the state-of-the art in mixed-criticality systems for space applications. The computing platform for the case study is described in section 4. Section 5 summarises the safety-related requirements of the satellite software. The safety concept for the qualification of the software is defined in section 6. Finally, section 7 contains some conclusions and hints for future work.

2. THE UPMSAT-2 SATELLITE

UPMSat2 is a university project aimed at building an experimental micro-satellite that can be used as a platform for experimenting with space technologies and as a support for teaching. The project is being carried out by a multidisciplinary team including several research groups at UPM with collaboration of a number of industrial companies. The satellite is expected to be launched in the last quarter of 2016 (Alonso et al., 2103).

The satellite has a single on-board computer (OBC) that is in charge of all platform management, communications, and control tasks in the satellite. The computer is complemented with a number of analog, digital, and serial interfaces connected to the satellite sensors, actuators, and the radio equipment providing communication links with ground support stations. The payload of the satellite consists of a number of experiments that have been proposed by industrial companies and research groups in order to test new hardware devices or new algorithms to be used in space systems.

The on-board software system is composed of a number of subsystems, each of them providing a different set of functions. Figure 1 displays the current architectural design. Its main components are:

- *Manager.* This subsystem drives the on-board software operation, controls the operation mode of the satellite, and performs the functions commonly grouped as *on-board data handling (OBDH)* (Fortescue et al., 2011).
- *Platform.* This module is in charge of acquiring platform (*housekeeping*) data from voltage and temperature sensors, and checking their validity.

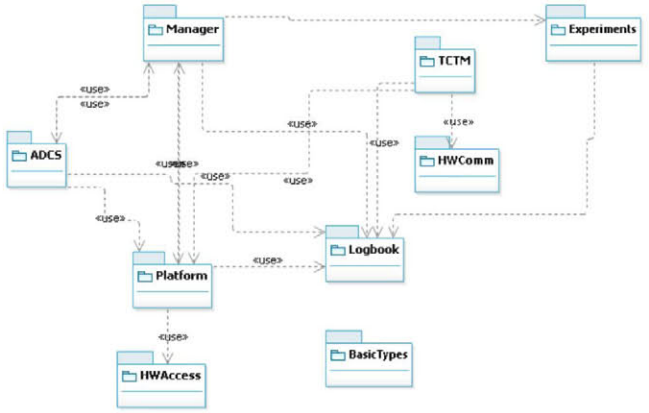

Fig. 1. Architectural design of the UPMSat2 software.

- *Attitude determination and control subsystem (ADCS).* This subsystem is in charge of controlling the attitude, i.e. the orientation of the satellite with respect to the Earth.
- *Hardware access.* This module encapsulates access to the hardware devices.
- *Experiments.* This subsystem controls the execution of the experiments, the collection of data resulting from them, and the expedition of telemetry packets with the results to be analysed on ground.
- *Telecommand and telemetry (TC/TM).* This module is in charge of communication with the ground stations.
- *Communications hardware (HWComm).* This module encapsulates access to the radio hardware.
- *Logbook.* This module provides non-volatile storage to the other modules.

The software is written in a safe subset of Ada 2005 restricted as per the Ravenscar profile (ARM05). The GNATforLEON compilation chain (Ruiz, 2005) is being used, together with additional tools supporting testing and some kinds of static analysis in order to facilitate validation. The application software runs on top of the GNAT run-time library (GNARL) and the ORK+ real-time kernel, which is developed and maintained by the UPM team (de la Puente et al., 2000).

The original design has been modified to use a mixed-criticality partitioned architecture (Salazar et al., 2014), for validating the developed technology and experimenting with this type of systems. This approach provides a better use of resources by separating the platform software components from the payload applications, and thus enabling them to be validated independently.

3. MIXED CRITICALITY SYSTEMS

3.1 Definition

In many domains, including aerospace systems, there is a trend to integrate multiple functions with different criticality levels on a shared embedded computing platform (Baumann et al., 2011). This kind of systems are called mixed-criticality systems.

Although mixed-criticality systems can be implemented in different ways (Burns and Davis, 2013), time and space partitioning (TSP) is the most promising approach from an industrial point of view (Crespo et al., 2014). Partitions provide functional separation and fault containment among applications or subsystems, in order to prevent any application from causing a failure in another application.

The basic idea behind this approach is *virtualization*, a technique that provides a virtual machine to each partition, on which applications can run isolated from each other. Virtualization is implemented by a software layer known as a *hypervisor* or *virtualization kernel*. In real time embedded applications, predictability and efficiency are important requirements. The virtualization techniques used to achieve temporal and spatial isolation jointly with real-time constraints require strict design methods and efficient solutions to guarantee correct system behaviour. Therefore, hypervisors for real-time mixed-criticality systems must provide timing guarantees in addition to special and temporal isolation between partitions.

3.2 Aerospace systems

The avionics industry has widely adopted the Integrated Modular Avionics (IMA) architecture as defined in the ARINC 653 standard (ARINC). In the space domain, ESA has recently launched the IMA for Space project in order to define a TSP reference architecture (Windsor and Hjortnaes, 2009). Several studies have been carried out in this framework(Windsor et al., 2011) with promising results. A recent study on porting the EagleEye reference mission software to a partitioned architecture (Bos et al., 2013) has demonstrated the feasibility of the approach on a real on-board software system.

3.3 Multicore platforms

The trend towards using multicore processors in embedded systems adds significant complexity to the design of mixed-criticality systems (Fuchsen, 2010). Using multicore processors in safety-critical systems raises some issues that have to be solved (Kinnan, 2009). The most important ones are related to spatial and temporal isolation at the hardware level. Spatial isolation can commonly be achieved using memory management units, but temporal isolation is more difficult to attain. The main reason for it is the use of shared hardware resources by different processor cores, which produces temporal interference between software partitions (Kotaba et al., 2013; Nélis et al., 2013). If the interference can be bounded, though, its effect on the temporal behaviour of the system can be analysed and incorporated to response time analysis techniques.

4. EXECUTION PLATFORM

4.1 On-board computer

The mixed-critical version of the software system has been designed to run on a multicore hardware platform, in order to enhance the performance of the system and experiment with this kind of architecture.

The modified OBC hardware is based on a dual-core LEON3 processor (Gaisler), built on a Xilinx Spartan

FPGA. A commercial development kit, including an FPGA board and an expansion board has been used to develop the new platform. [3]

4.2 Hypervisor layer

A hypervisor is a software layer that implements partitions using virtualization. Therefore, a hypervisor can be used as a separation kernel in a partitioned architecture. The main function of the hypervisor is to provide time and space separation between partitions, in such a way that partitions can be validated independently, provided that the hypervisor itself has been qualified to the highest criticality level of all the applications or subsystems running in the partitions.

A multicore version of the XtratruM hypervisor (Masmano et al., 2010) has been used to build the partitioned version of the UPMSat2 software. The hypervisor runs directly on the hardware, creating several abstract runtime environments (partitions) where different applications or subsystems can be executed. Partitions are executed in temporal and spatial isolation from each other.

XtratuM provides virtualization services to the partitions. It is executed in supervisor processor mode, and virtualizes the CPU, memory, interrupts, as well as some specific peripherals.

4.3 Operating system layer

The Open Ravenscar real-time Kernel (ORK+, de la Puente et al. 2000) has been used for the operating system layer as it provides full conformance for the real-time tasking model defined by the Ada Ravenscar profile. The profile restricts the tasking model to a static set of periodic and sporadic tasks with a fixed-priority pre-emptive scheduling method, communicating through shared data objects protected for mutually exclusive access (Burns et al., 2003). The restrictions enable static response-time analysis to be used to assess the temporal behaviour of real-time system on monoprocessor platforms. The kernel has been extended to run on XtratuM on a multicore LEON3 architecture (Esquinas et al., 2011).

4.4 Qualification considerations

The XtratuM multicore version derives from that used in the IMA-SP project (Windsor et al., 2011). Further verification and validation activities have been carried out to ensure compliance with the ECSS standards (ECSS-E40; ECSS).

The GNAT Pro runtime environment used in this study has evolved from the AdaCore [4] GNAT Pro Safety-Critical development environment, which is based on the original ORK real-time kernel (Ruiz, 2005). The run-time library included in it has been qualified to criticality level B of the ESA standards.

[3] `http://www.xilinx.com/products/boards-and-kits/AES-S6IEK-LX150T-G.htm`

[4] `www.adacore.com`

5. SAFETY REQUIREMENTS

The rationale for the safety requirements of the system comes from the ECSS safety and software engineering standards ECSS-Q-ST-40C (ECSS-Q40) and ECSS-E-ST-40 (ECSS-E40), and from the requirements baseline for the UPMSat-2 Software (de la Puente et al., 2014a). A selection of such requirements follows.

(1) Faults detected during the operation of the system will be recorded as significant events. Subsequent actions may include mode changes or hardware reset of the on-board computer.
(2) Watchdog Timers (WDT) will be used to detect possible failures of the computer system and the partitions. If the timer expires, a hardware reset of the computer or a reset of the required partitions will be issued.
(3) The system must monitor the state of the satellite platform by a periodical sampling of representative variables of temperature and voltage at select points of the satellite (housekeeping data). The measured values will be checked for validity, and values outside their valid range will be reported.
(4) The state of the satellite shall be transmitted to the ground segment for analysis and event or failure handling.
(5) The ground segment shall send telecommands to the satellite to request operations such as changing the operation mode, setting the state of hardware devices, or updating the validity ranges of sensor values.
(6) The attitude control output will be validated in order to ensure that proper values are sent to the actuators. In case of values out of range, the event will be reported.
(7) The system operates in a set of modes. Transitions between modes are triggered by events detected in the monitoring functions, finalization of operations, timer expiration or ground commands. The functions to be executed on each mode are adapted to the satellite state. The software shall ensure that mode transitions are carried out as specified.
(8) The system shall change to safe mode in case of low battery or fault detection, or upon reception of a telecommand. System operation during this mode is degraded.
(9) The system shall change to beacon mode if communication with ground is lost. System operation is limited to transmitting periodic telemetry messages, until communication is recovered.
(10) Telecommands will carry digital identification codes in order to ensure that a valid ground station has issued them.
(11) The system must guarantee time requirements.
(12) Software failures cannot be propagated in an uncontrolled way.

6. SAFETY CONCEPT

6.1 General approach

The UPMSat2 project is aimed at experimenting with space technologies and supporting teaching. The project is led by IDR (Ignacio da Riva Institute at the Universidad

Politécnica de Madrid), which has carried out most of the activities related to the design and building of the satellite, including hazard and risk analysis of the overall system. The design of the different subsystems is aimed, among other goals, at eliminating or minimizing the most severe hazards, until the associated risks are acceptable.

It is important to note that the nature and aims of the UPMSat2 project are directed towards building a micro-satellite, with limited size and budget. This implies that the acceptable risks are more severe than for other types of satellites. In particular, the size of the satellite precludes the extensive use of redundancy for the most critical subsystems, even in the case that their failure may imply the loss of the mission. For instance, neither the satellite antenna and communications equipment, nor the computer system are replicated. On the other hand, there are some replicated components. This is the case for the solar panels. Their capacity is larger than needed in the nominal state, in order to deal with some kinds of failures. In case of failure of one of these elements, it would simply cease to provide its functionality.

The same kind of analysis has been carried out for the software system, where its most critical components are classified as category B, i.e. "software that if not executed, or if not correctly executed or whose anomalous behaviour can cause or contribute to a system failure resulting in critical consequences, usually implying the loss of the mission" (ECSS-Q40). It can be discussed if this is suitable, given the limited fault tolerance approach of the overall satellite design, as it implies higher development costs and complexity. However, the purpose of UPMSat2 as a research and education vehicle fully justifies this decision.

In the rest of this section, a multicore partitioned version of the UPMSat2 software is described, and the safety concept for two particular aspects of the on-board computing system is discussed.

6.2 Architecture of the partitioned system

In order to assess the feasibility of qualifying a satellite computer system with a multi-core partitioned mixed-criticality system, a new version of the system has been designed (de la Puente et al., 2014b). As explained in section 4, the hardware platform has been changed to a dual-core LEON3 processor with shared memory, with an instance of the XtratuM hypervisor running on it.

The application software has been divided into several partitions (see figure 2), according to general software engineering practices and to the following principles:

- *Criticality*: Modules with different criticality level are located in different partitions. For example, the payload experiments are considered less critical than the modules managing the satellite platform.
- *Separation of concerns*: Modules with related functionality are located in the same partition.
- *Reuse*: Some modules have a greater potential for reuse, as is the case for the TC/TM module that can be based on ECSS communication standards. If software for this modules is reused, it can be allocated to a dedicated partition.

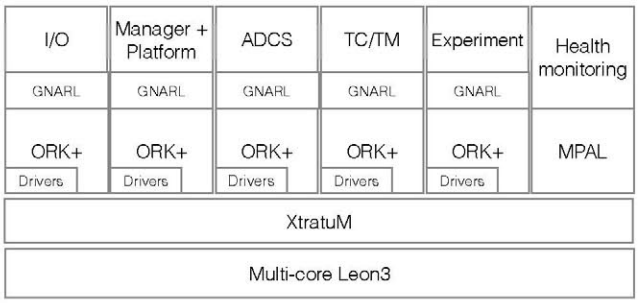

Fig. 2. Partitioned architecture of the space case study system.

- *Hardware isolation*: Modules that directly depend on the hardware are located in a specific I/O partition handling all the hardware interfaces.
- *Safety*: A specific partition can be allocated to so-called *health monitoring* functions.

As shown in figure 2, there is a copy of the operating system (ORK+) and the Ada runtime (GNARL) in each application partition, on top of which the application software is executed.

The software for the partitioned architecture has been developed using the MultiPARTES toolset (Salazar et al., 2013). The tools generate code skeletons of software components, into which functional code is inserted. Scheduling plans for the partitions are generated with the partitioning tool included in the toolset.

6.3 Safety concept for the satellite software

The safety concept for the on-board software system is based on the following principles (see section 2 for the names of the subsystems):

(1) The manager is the main responsible for driving the on-board computer operation. It changes the operation mode, in response to faults, events and telecommands.
(2) The TC/TM module forwards telecommands to the manager, which either handles them directly, or redirects them to another software module.
(3) The platform module executes the housekeeping functions. It monitors the state of the satellite by sampling housekeeping sensors. Then, it checks if sensors are working properly, and if the acquired values are within their validity ranges. If these conditions are not met, it signals a fault or event to the manager. Finally, it enqueues a message with housekeeping data to be sent to ground.
(4) The ADCS module checks if the signal to be sent to the actuators is within its validity margins. Otherwise, it notifies this event to the manager.
(5) The implementation of modules ensures that execution errors are contained and are not propagated in an uncontrolled way. They are handled internally, or otherwise the partition is stopped.
(6) Watchdog timers (WDT) are used for detecting failures in the proper operation of the partitions and the system. A virtual WDT is managed by the hypervisor for each partition. If a WDT expires, the associated partition is restarted. The time required for this oper-

ation is fast enough to avoid a global system reboot. This option is used in case of major failures.
(7) A hardware WDT is used for detecting possible failures of the hypervisor. If this timer expires, the computer is reset.
(8) Telecommands will be either signed or encrypted in order to ensure that a valid ground station issues them. The TC/TM module checks their validity before forwarding them to the manager.

Some of these principles will now be described in more detail.

Fault detection, isolation, and recovery

The original UPMSat2 requirements baseline states that faults detected may raise mode changes or hardware resets. In addition, a hardware watchdog timer (WDT) will be used to detect possible failures of the computer system. If this timer expires, a hardware reset will be used to restart the on-board computer (de la Puente et al., 2014a).

This requirement is applied to the partitioned version of the satellite software. The hardware WDT is handled by the hypervisor, which is in charge of periodically arming the timer. If it reaches zero, the computer is restarted.

Special consideration has to be taken for each of the partitions. Although the same approach based on hardware WDT could be used for each partition, in practice it is difficult to achieve. Thus, virtual WDTs are used for the partitions. Each partition is in charge of setting its own virtual WDT. If the timer reaches zero, a trap is raised, which is captured by the hypervisor, and a predefined handler that restarts the partition is executed. The same approach is used for dealing with traps caused by other types of software errors.

Satellite housekeeping: failure detection and handling

The ECSS-Q-ST-40C standard (ECSS-Q40) defines a set of special procedures for dealing with hazards that are not minimized or controlled by fault tolerance or specific devices. This is the case for most of the satellite equipment, as mentioned above. In this case, it is required to provide real-time monitoring, hazard detection and safing systems for hazard control. In addition, with respect to hazard detection, signalling and safing, the standard indicates that the system design shall provide the capability for detecting failures that result in degradation or fault tolerance. Such failures have to be notified to the mission operators on ground.

The satellite includes a number of sensors for measuring temperatures, voltages and currents of different system components. The software specification requires that housekeeping data are acquired periodically and checked for validity. Values out of range are reported to ground. In addition, in some cases the on-board software can take automatic actions for dealing with these values. For instance, if the battery level is too low, the system is required to change to safe mode. As another example, the attitude control algorithm can cease temporarily to use the output of a malfunctioning sensor, as described above.

Housekeeping data are sent to ground, where they are analysed. The system operators have some means for dealing with invalid sensor values. For example, they can issue telecommands for changing the validity ranges or mark a sensor as faulty.

6.4 Software development

The software development process follows the ECSS standards in order to ensure that the system requirements are met and the software behaves properly. This includes the following aspects:

- Documentation must be written according to ECSS standards.
- Reviews by team members not directly related with the development process must be carried out. The reviews will include requirements traceability, review of the development artefacts, and testing procedures.
- Propagation of software failures is controlled. Every software module includes handlers for avoiding uncontrolled exception propagation. Task termination, forbidden by the Ravenscar profile, will raise a trap that will be handled by the hypervisor. Finally, the isolation of applications as enforced by the hypervisor ensures that failures in an application will not affect others in an uncontrolled way.
- An object-oriented design approach will be used, using UML. The detailed design will map objects onto Ada entities.
- Static analysis will be used to verify robustness with respect to errors that are difficult to detect at runtime, such as division by zero, etc.
- The testing procedures should ensure the fulfilment of coverage metrics for criticality category B.
- Schedulability analysis will be used at different development phases to check the fulfilment of time requirements.

7. CONCLUSIONS

A safety concept for embedded real-time systems running on partitioned multicore architectures has been presented. The safety concept has been applied to the on-board software system of an experimental satellite project. The software system has components with different criticality levels, and the safety requirements for some of the most critical components have been analysed independently of other components.

This approach opens new alternatives to the development of mixed-criticality on-board systems, and is compatible with the software standards used in the European space industry. Future work includes checking the approach with an independent software assurance company in order to assess the possibility of qualifying satellite systems using partitioned multicore architectures in an industrial framework.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the collaboration of the MultiPARTES partners, especially the XtratuM team at UPV and FentISS, and Ikerlan-IK4.

The UPMSat2 project is led by the *Ignacio da Riva* Institute (IDR) of the Technical University of Madrid. [5] .

REFERENCES

Alonso, A., Salazar, E., and de la Puente, J.A. (2103). Design of on-board software for an experimental satellite. In *Jornadas de Tiempo Real — JTR-2013*. URL `www.dit.upm.es/str/papers/pdf/alonso&13a.pdf`.

ARINC (2003). *Avionics Application Software Standard Interface — ARINC Specification 653-1*. ARINC.

ARM05 (2007). *ISO/IEC 8652:1995(E)/ TC1(2000)/ AMD1(2007): Information Technology — Programming Languages — Ada*.

Baumann, C., Blasum, H., Bormer, T., and Tverdyshev, T. (2011). Proving memory separation in a microkernel by code level verification. In *1st International Workshop on Architectures and Applications for Mixed-Criticality Systems (AMICS 2011)*.

Bos, V., Mendham, P., Kauppinen, P., Holsti, N., Crespo, A., Masmano, M., de la Puente, J., and Zamorano, J. (2013). Time and space partitioning the EagleEye Reference Mission. In *Data Systems in Aerospace — DASIA 2013*. Porto, Portugal.

Burns, A. and Davis, R. (2013). Mixed criticality systems — A review. Technical report, University of York. URL `http://www-users.cs.york.ac.uk/~burns/review.pdf`.

Burns, A., Dobbing, B., and Vardanega, T. (2003). Guide for the use of the Ada Ravenscar profile in high integrity systems. Technical Report YCS-2003-348, University of York.

Crespo, A., Alonso, A., Marcos, M., de la Puente, J.A., and Balbastre, P. (2014). Mixed-criticality in control systems. In E. Boje and X. Xia (eds.), *Proc. 19th IFAC World Congres*, 12261–12271. IFAC-PapersOnLine.

de la Puente, J.A., Ruiz, J.F., and Zamorano, J. (2000). An open Ravenscar real-time kernel for GNAT. In H.B. Keller and E. Plödereder (eds.), *Reliable Software Technologies — Ada-Europe 2000*, number 1845 in LNCS, 5–15. Springer-Verlag.

de la Puente, J.A., Zamorano, J., and Alonso, A. (2014a). UPMSAT-2 software. requirements baseline — Software System Specification. Technical report, Universidad Politécnica de Madrid. Version 2.2.

de la Puente, J.A., Zamorano, J., Alonso, A., Garrido, J., Salazar, E., and de Miguel, M.A. (2014b). Experience in spacecraft on-board software development. *Ada User Journal*, 35(1), 55–60.

ECSS (2009). *ECSS-Q-ST-80C Space Product Assurance — Software Product Assurance*. European Cooperation for Space Standardization. Available from ESA.

ECSS-E40 (2009). *ECSS-E-ST-40C Space engineering — Software*. European Cooperation for Space Standardization. Available from ESA.

ECSS-Q40 (2009). *ECSS-Q-ST-40C Space product assurance — Safety*. European Cooperation for Space Standardization. Available from ESA.

Esquinas, A., Zamorano, J., de la Puente, J.A., Masmano, M., Ripoll, I., and Crespo, A. (2011). ORK+/XtratuM: An open partitioning platform for Ada. In A. Romanovsky and T. Vardanega (eds.), *Reliable Software Technologies — Ada-Europe 2011*, number 6652 in LNCS, 160–173. Springer-Verlag.

Fortescue, P., Swinerd, G., and Stark, J. (2011). *Spacecraft Systems Engineering*. Wiley, 4 edition.

Fuchsen, R. (2010). How to address certification for multicore based IMA platforms: Current status and potential solutions. In *IEEE/AIAA 29th Digital Avionics Systems Conference (DASC)*.

Gaisler (2012). *LEON3 - High-performance SPARC V8 32-bit Processor. GRLIB IP Core User's Manual*. Gaisler Research.

Kinnan, L. (2009). Use of multicore processors in avionics systems and its potential impact on implementation and certification. In *IEEE 28th Digital Avionics System Conference*.

Kotaba, O., Nowotsch, J., Paulitsch, M., Petters, S., and Theilingx, H. (2013). Multicore in real-time systems – temporal isolation challenges due to shared resources. In *Workshop on Industry-Driven Approaches for Cost-effective Certification of Safety-Critical, Mixed-Criticality Systems (WICERT)*.

Masmano, M., Ripoll, I., Crespo, A., and Peiró, S. (2010). XtratuM for LEON3: An opensource hypervisor for high-integrity systems. In *Embedded Real Time Software and Systems — ERTS2 2010*. Toulouse (France).

Nélis, V., Pinho, L.M., Bertogna, M., Vargas, R., Yomsi, P.M., Fonseca, J., Quiñones, E., and Marongiu, A. (2013). The challenge of time-predictability in modern many-core architectures. Technical report, CISTER, ISEP-IPP. CISTER-TR-140624.

Ruiz, J.F. (2005). GNAT Pro for on-board mission-critical space applications. In T. Vardanega and A. Wellings (eds.), *Reliable Software Technologies — Ada-Europe 2005*, volume 3555 of *LNCS*. Springer-Verlag.

Salazar, E., Alonso, A., de Miguel, M.A., and de la Puente, J.A. (2013). A model-based framework for developing real-time safety Ada systems. In H. Keller, E. Plödereder, P. Dencker, and H. Klenk (eds.), *Reliable Software Technologies - Ada-Europe 2013*, volume 7896 of *Lecture Notes in Computer Science*, 127–142. Springer-Verlag.

Salazar, E., Alonso, A., and Garrido, J. (2014). Mixed-criticality design of a satellite software system. In E. Boje and X. Xia (eds.), *Proc. 19th IFAC World Congres*, 12278–12283. IFAC-PapersOnLine.

Windsor, J., Deredempt, M., and De-Ferluc, R. (2011). Integrated modular avionics for spacecraft —– User requirements, architecture and role definition. In *IEEE/AIAA 30th Digital Avionics Systems Conference (DASC)*.

Windsor, J. and Hjortnaes, K. (2009). Time and space partitioning in spacecraft avionics. In *Third IEEE International Conference on Space Mission Challenges for Information Technology*.

[5] `www.idr.upm.es`